Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №8

«Виконання операцій з плаваючою точкою та вивчення команд x87 FPU»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

Мета

Навчитися програмувати операції з плаваючою точкою на асемблері.

Завдання

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab8.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути головний файл main8.asm та інші модулі (за необхідності).

3. У цьому проекті кожний модуль може окремо компілюватися.

4. Скомпілювати вихідний текст і отримати виконуємий файл програми.

5. Перевірити роботу програми. Налагодити програму.

6. Отримати результати – файл числових значень згідно варіанту завдання.

7. Проаналізувати та прокоментувати результати, вихідний текст та дизасембльований машинний код програми.

Мій варіант

4	X_1, X_2 = Рішення системи лінійних рівнянь 2-го порядку	64-бітовий

Оскільки у файлі до лабораторної роботи всього 21 варіант, а я 25 у списку, то взяв за свій варіант 4 завдання, оскільки 21 + 4 = 25.

Текст програми

main8.asm

```
.586
.model flat, stdcall

option casemap : none; розрізнювати великі та маленькі букви
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc

include module.inc
include longop.inc

includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib

.data
CaptionHello db 'Лабораторна робота № 8', 0
TextHello db 'Автор: Шендріков Євгеній', 13, 10, 'Група: ІО-82', 13, 10, "Номер у списку: 25",
13, 10, "Рік: 2020", 0

Caption1 db "X1", 0
Caption2 db "X2", 0
```

```
;введення коефіцієнтів рівняння типу
;A11*X1 + A12*X2 = B1
;A21*X1 + A22*X2 = B2
valueA11 dd 2.3
valueA22 dd 1.2
valueA21 dd 3.4
valueA12 dd 0.6
valueB1 dd 4.1
valueB2 dd 2.6

buffOperand1 dd ?
buffOperand2 dd ?
buffOperand3 dd ?

resultX1 dd 0
resultX2 dd 0
X1Text dd 100 dup(0)
X2Text dd 100 dup(0)

.code
main:
	invoke MessageBoxA, 0, ADDR TextHello, ADDR CaptionHello, 0
	;рахуємо детермінант системи
	fld valueA11
	fmul valueA22
	fld valueA21
	fmul valueA12
	fsub
	fst buffOperand1

	;рахуємо значення X1
	fld valueB1
	fmul valueA22
	fld valueB2
	fmul valueA12
	fsub
	fst buffOperand2
	fld buffOperand2
	fdiv buffOperand1
	fstp resultX1

	;рахуємо значення X2
	fld valueB2
	fmul valueA11
	fld valueB1
	fmul valueA21
	fsub
	fst buffOperand3
	fld buffOperand3
	fdiv buffOperand1
	fstp resultX2

	push offset X1Text
	push  resultX1
	call FloatDec_MY
	invoke MessageBoxA, 0, ADDR X1Text, ADDR Caption1, 0

	push offset X2Text
	push  resultX2
	call FloatDec_MY
	invoke MessageBoxA, 0, ADDR X2Text, ADDR Caption2, 0
	invoke ExitProcess, 0
end main
```

longop.asm

```
.586
.model flat, c
.code

FloatDec_MY proc
	push ebp
	mov ebp, esp
	mov esi, [ebp + 8];esi 32 бітове число
	mov edi, [ebp + 12];edi адреса текстового буферу

	;вставка знаку числа
	mov eax, esi
	and eax, 80000000h
	cmp eax, 0
	je @end_sign
			mov byte ptr[edi], 45
			inc edi
	@end_sign:
	mov ecx, edi

	;перевірка e
	mov eax, esi
	and eax, 7F800000h
	shr eax, 23

	cmp eax, 0
	jne @next
		mov byte ptr[edi], 48
		jmp @endproc

	@next:
		cmp eax, 0FFh
		jne @next2
		mov byte ptr[edi], 78
		mov byte ptr[edi + 1], 65
		mov byte ptr[edi + 2], 78
	jmp @endproc

	@next2:
	sub eax, 7Fh

	;перевірка E
	cmp eax, 0
	jge @next3
			mov byte ptr[edi], 48
			inc ecx
			mov ebx, esi
			and ebx, 7FFFFFh ;мантиса
			add ebx, 800000h ;прихована одиниця в 24 розр.
			mov edx, 0FFFFFFFFh ;-1
			imul edx
			mov edx, ecx
			mov ecx, eax
			shr ebx, cl
			mov ecx, edx
			jmp @fraction

	@next3:
		jg @next4
		mov byte ptr[edi], 49
```

```
        inc ecx
        mov ebx, esi
        and ebx, 7FFFFFh
        jmp @fraction

@next4:
        push ecx

        ;пошук цілої частини
        mov ecx, 23
        sub ecx, eax ;експонента (позиція від якої накладати маску)
        push ecx
        mov eax, esi
        and eax, 7FFFFFh
        add eax, 800000h ;прихована одиниця в 24 розр.
        xor ebx, ebx
        mov ebx, 1
        shl ebx, cl
        mov edx, ebx

@mask:
        inc cl
        shl ebx, 1
        add ebx, edx
        cmp cl, 24
jne @mask

mov edx, eax
and edx, ebx ;ціла частина

mov ebx, eax
sub ebx, edx ;дробова частина

pop ecx
shr edx, cl  ;здвиг цілої частини до 0 розряду

mov eax, 23
sub eax, ecx
mov ecx, eax
shl ebx, cl  ;здвиг дробової частини до 23 розряду

mov eax, edx
pop ecx
push ebx
mov ebx, 10
@full_part: ;обрахунок цілої частини
        xor edx, edx
        div ebx
        add edx, 48
        mov byte ptr[ecx], dl
        inc ecx
        cmp eax, 0
jne @full_part

mov eax, ecx
dec eax
@reverse:
        xor edx, edx
        mov dh, byte ptr[eax]
        mov dl, byte ptr[edi]
        mov byte ptr[eax], dl
        mov byte ptr[edi], dh
        inc edi
```

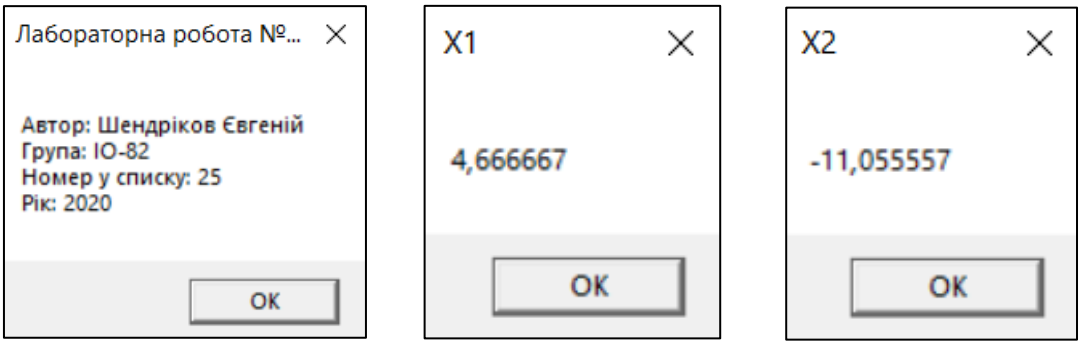
```
		dec eax
		cmp edi, eax
	jl @reverse

	pop ebx

	;пошук дробової частини в ebx 23 разр. дробу
	@fraction:
		mov byte ptr[ecx], 44
		inc ecx
		mov ax, 6
	@cycle:
		shl ebx, 1
		mov edx, ebx
		shl edx, 2
		add ebx, edx

		mov edx, ebx
		and edx, 0FF800000h
		shr edx, 23
		add dl, 48
		mov [ecx], dl
		and ebx, 7FFFFFh
		inc ecx
		dec ax
		cmp ax, 0
	jne @cycle
	@endproc:
		pop ebp
		ret 8
FloatDec_MY endp

end
```

module.asm

```
.386
.model flat, c
.code

;процедура StrHex_MY записує текст шістнадцятькового коду
;перший параметр - адреса буфера результату (рядка символів)
;другий параметр - адреса числа
;третій параметр - розрядність числа у бітах (має бути кратна 8)
StrHex_MY proc
	push ebp
	mov ebp,esp
	mov ecx, [ebp+8] ;кількість бітів числа
	cmp ecx, 0
	jle @exitp
	shr ecx, 3 ;кількість байтів числа
	mov esi, [ebp+12] ;адреса числа
	mov ebx, [ebp+16] ;адреса буфера результату
@cycle:
	mov dl, byte ptr[esi+ecx-1] ;байт числа - це дві hex-цифри
	mov al, dl
	shr al, 4 ;старша цифра
	call HexSymbol_MY
	mov byte ptr[ebx], al
	mov al, dl ;молодша цифра
	call HexSymbol_MY
	mov byte ptr[ebx+1], al
	mov eax, ecx
```

```
	cmp eax, 4
	jle @next
	dec eax
	and eax, 3 ;проміжок розділює групи по вісім цифр
	cmp al, 0
	jne @next
	mov byte ptr[ebx+2], 32 ;код символа проміжку
	inc ebx
@next:
	add ebx, 2
	dec ecx
	jnz @cycle
	mov byte ptr[ebx], 0 ;рядок закінчується нулем
@exitp:
	pop ebp
	ret 12

StrHex_MY endp

;ця процедура обчислює код hex-цифри
;параметр - значення AL
;результат -> AL
HexSymbol_MY proc
	and al, 0Fh
	add al, 48 ;так можна тільки для цифр 0-9
	cmp al, 58
	jl @exitp
	add al, 7 ;для цифр A,B,C,D,E,F
@exitp:
	ret
HexSymbol_MY endp


;ця процедура записує 8 символів HEX коду числа
;перший параметр - 32-бітове число
;другий параметр - адреса буфера тексту
DwordToStrHex proc
push ebp
mov ebp,esp
mov ebx,[ebp+8] ;другий параметр
mov edx,[ebp+12] ;перший параметр
xor eax,eax
mov edi,7
@next:
mov al,dl
and al,0Fh ;виділяємо одну шістнадцяткову цифру
add ax,48 ;так можна тільки для цифр 0-9
cmp ax,58
jl @store
add ax,7 ;для цифр A,B,C,D,E,F
@store:
mov [ebx+edi],al
shr edx,4
dec edi
cmp edi,0
jge @next
pop ebp
ret 8
DwordToStrHex endp

end
```

Результати роботи програми



Аналіз результатів

Для розв'язання поставленої задачі, тобто знайди розв'язок СЛАР з двома невідомими, я обрав методом Крамера. Розв'язання рівнянь з числами, які я використав у програмі має вигляд:

$$\begin{cases} A_{11} * X_1 + A_{12} * X_2 = B_1 \\ A_{21} * X_1 + A_{22} * X_2 = B_2 \end{cases}$$

$$\begin{cases} 2.3X_1 + 0.6X_2 = 4.1 \\ 3.4X_1 + 1.2X_2 = 2.6 \end{cases}$$

$$\Delta = \begin{vmatrix} 2.3 & 0.6 \\ 3.4 & 1.2 \end{vmatrix} = 0.72$$

$$\Delta_1 = \begin{vmatrix} 4.1 & 0.6 \\ 2.6 & 1.2 \end{vmatrix} = 3.36$$

$$\Delta_2 = \begin{vmatrix} 2.3 & 4.1 \\ 3.4 & 2.6 \end{vmatrix} = -7.96$$

$$X_1 = \frac{\Delta_1}{\Delta} = 4.666667 \qquad X_2 = \frac{\Delta_2}{\Delta} = -11.055557$$

Результати програми повністю збігаються з теоретичними даними. Програма працює вірно. У самій програмі розв'язання виконується за кінцевими формулами розрахунку: $X_1 = \frac{B_1 * A_{22} - B_2 * A_{12}}{A_{11} * A_{22} - A_{12} * A_{22}}$; $X_2 = \frac{B_2 * A_{11} - B_1 * A_{21}}{A_{11} * A_{22} - A_{12} * A_{22}}$.

Спочатку виконується обрахунок знаменника дробів. Якщо детермінант матриці дорівнює 0, то у такому випадку система не має розв'язку, тому виводяться нулі замість кожного з ікс. Якщо детермінант не нуль, тоді обрахунок інших частин і нарешті самих невідомих. Обрахунок здійснюється за допомогою команд x87 FPU, а саме: FMUL, FSUB, FDIV.

Висновок

Під час виконання лабораторної роботи було вдосконалено знання роботи з модулями і здобуто навичку ділити числа з плаваючою точкою за допомогою FPU x87, розв’язуючи завдання лабораторної роботи. Кінцева мета роботи досягнута.